Factorization in Deep Neural Networks - Part 2

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Course organisation

Sessions

1. Deep Learning and Transfer Learning,
2. Quantization,
3. Pruning,
4. Factorization,
5. Fact. pt.2 : Operators and Architectures,
6. Distillation,
7. Embedded Software and Hardware for DL.
8. Presentations for challenge.

Course organisation

Sessions

1. Deep Learning and Transfer Learning,
2. Quantization,
3. Pruning,
4. Factorization,
5. Fact. pt.2 : Operators and Architectures,
6. Distillation,
7. Embedded Software and Hardware for DL.
8. Presentations for challenge.

General principle

Complexity of 2D Convolutions

$N_{ops} = H_{out} \cdot W_{out} \cdot h \cdot w \cdot D_{in} \cdot D_{out}$
with kernel size (h, w), D_{in} the number of input feature maps, D_{out} the number of output feature maps of height H_{out} and width W_{out}.

To reduce the number of parameters, we can :

- Reduce the size of kernels
- Reduce the number of feature maps

Different strategies :

- Decompose kernels
- Depthwise Separable Convolutions
- Grouped Convolutions

General principle

Complexity of 2D Convolutions

$N_{ops} = H_{out} \cdot W_{out} \cdot h \cdot w \cdot D_{in} \cdot D_{out}$
with kernel size (h, w), D_{in} the number of input feature maps, D_{out} the number of output feature maps of height H_{out} and width W_{out}.

To reduce the number of parameters, we can :

- Reduce the size of kernels
- Reduce the number of feature maps

Different strategies :

- Decompose kernels
- Depthwise Separable Convolutions
- Grouped Convolutions

General principle

Complexity of 2D Convolutions

$N_{ops} = H_{out} \cdot W_{out} \cdot h \cdot w \cdot D_{in} \cdot D_{out}$
with kernel size (h, w), D_{in} the number of input feature maps, D_{out} the number of output feature maps of height H_{out} and width W_{out}.

To reduce the number of parameters, we can :

- Reduce the size of kernels
- Reduce the number of feature maps

Different strategies :

- Decompose kernels
- Depthwise Separable Convolutions
- Grouped Convolutions

General principle

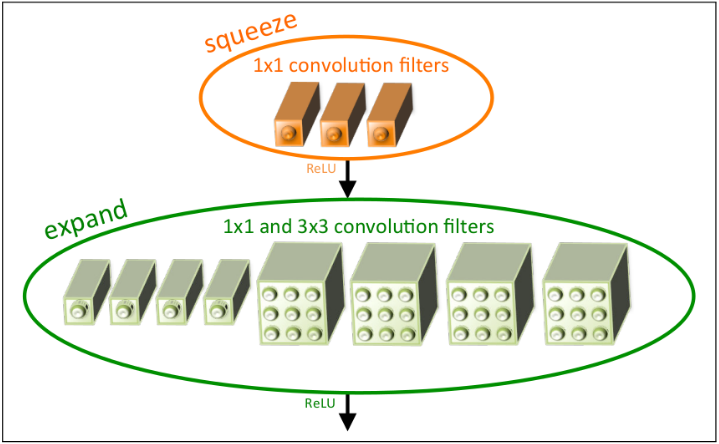
Complexity of 2D Convolutions

$N_{ops} = H_{out} \cdot W_{out} \cdot h \cdot w \cdot D_{in} \cdot D_{out}$
with kernel size (h, w), D_{in} the number of input feature maps, D_{out} the number of output feature maps of height H_{out} and width W_{out}.

Decomposing kernels

Assuming $D_{in} = D_{out}$, decompose (h, w) kernel by $(h, 1)$ and $(1, w)$:
$N_{ops} = h.1.D_{in}^2 + 1.w.D_{in}^2 = (h + w).D_{in}^2$
with kernel size (h, w), D_{in} input and out number of feature maps.

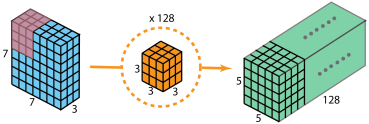
Introducing the Fire Module



Iandola et al. 2016, `https://arxiv.org/abs/1602.07360`

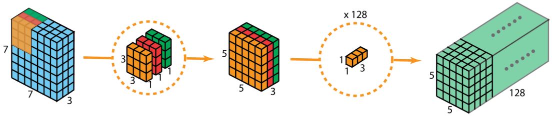
Depthwise serparable convolutions

Instead of learning parameters that recombine all input feature maps to compute each output feature map:



$$N_{mul} = H_{out} \cdot W_{out} \cdot h \cdot w \cdot D_{in} \cdot D_{out}$$
$$N_{mul} = 5 \cdot 5 \cdot 3 \cdot 3 \cdot 3 \cdot 128 = 86400$$

One can separate the operations into two steps:

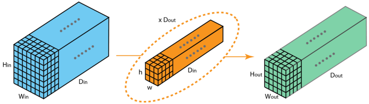


$$N_{mul} = H_{out} \cdot W_{out} \cdot h \cdot w \cdot D_{in} \cdot 1 + h \cdot w \cdot 1 \cdot 1 \cdot D_{in} \cdot D_{out}$$
$$N_{mul} = 5 \cdot 5 \cdot 3 \cdot 3 \cdot 3 \cdot 1 + 5 \cdot 5 \cdot 1 \cdot 1 \cdot 3 \cdot 128 = 10275$$

https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

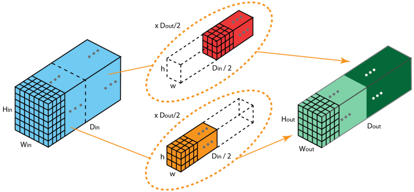
Grouped Convolutions

Instead of learning parameters that recombine all input feature maps to compute each output feature map:



$$N_{mul}^{N} = H_{out} \cdot W_{out} \cdot h \cdot w \cdot D_{in} \cdot D_{out}$$

One can divide the kernels into multiple groups:



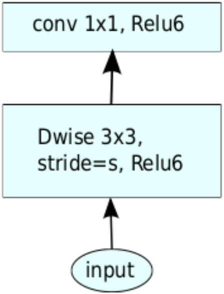
$$N_{mul}^{G} = H_{out} \cdot W_{out} \cdot h \cdot w \cdot \frac{D_{in}}{2} \cdot \frac{D_{out}}{2} + H_{out} \cdot W_{out} \cdot h \cdot w \cdot \frac{D_{in}}{2} \cdot \frac{D_{out}}{2}$$

$$N_{mul}^{G} = \frac{N_{mul}^{N}}{2}$$

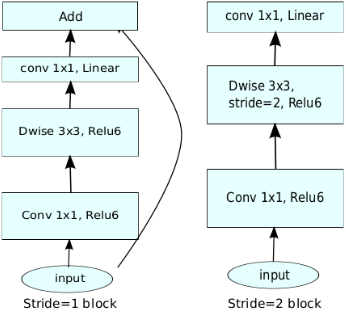
https://towardsdatascience.com/a-comprehensive-introduction-to-different-types-of-convolutions-in-deep-learning-669281e58215

MobileNet

MobileNetV1



MobileNetV2



`https://arxiv.org/abs/1704.04861` and `https://arxiv.org/abs/1801.04381`

MobileNet

Accuracy obtained on ImageNet

Network	Accuracy(%)	Params (M)
SqueezeNet	57.5	1.24
MobileNetV1	70.6	4.20
MobileNetV2	72.0	3.40

https://arxiv.org/abs/1704.04861 and https://arxiv.org/abs/1801.04381

Introducing Shift Attention Layer

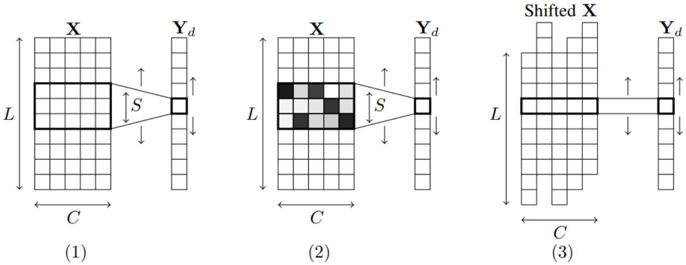


Figure 1: Overview of the proposed method: we depict here the computation for a single output feature map d, considering a 1d convolution and its associated shift version. Panel (1) represents a standard convolutional operation: the weight filter $\mathbf{W}_{d,\cdot,\cdot}$ containing SC weights is moved along the spatial dimension (L) of the input to produce each output in $\mathbf{Y}_d$. In panel (2), we depict the attention tensor $\mathbf{A}$ on top of the weight filter: the darker the cell, the most important the corresponding weight has been identified to be. At the end of the training process, $\mathbf{A}$ should contain only binary values with a single 1 per slice $\mathbf{A}_{d,c,\cdot}$. In panel (3), we depict the corresponding obtained shift layer: for each slice along the input feature maps (C), the cell with the highest attention is kept and the others are disregarded. As a consequence, the initial convolution with a kernel size S has been replaced by a convolution with a kernel size 1 on a shifted version of the input $\mathbf{X}$. As such, the resulting operation in panel (3) is exactly the same as the shift layer introduced in Wu et al. [2017], but here the shifts have been trained instead of being arbitrarily predetermined.

Hacene et al. 2019, `https://arxiv.org/abs/1905.12300`

Alternatives to Convolution

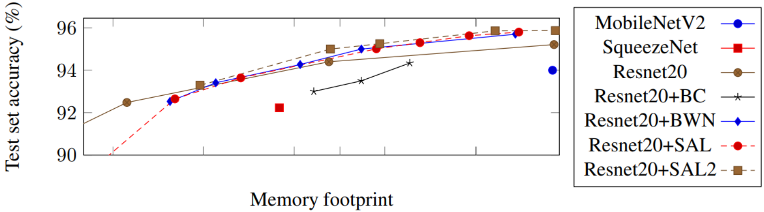


Figure 7: Evolution of accuracy when applying compression methods on different DNN architectures trained on CIFAR10.

Hacene et al. 2019, `https://arxiv.org/abs/1905.12300`